```
pragma solidity ^0.5.3;

interface IERC20 {

//totalSupply - it returns the initial quantity of rolled out tokens
    function totalSupply() external view returns (uint256);

//balanceOf - it returns the number of token hold by any particular
address
    function balanceOf(address _owner) external view returns (uint256
balance);

//transfer -  it is to trnasfer the token from one account to other
account
    function transfer(address _to, uint256 _value) external returns
(bool success);

//approve - owner approves a spender to use it's own token

    function approve(address _spender, uint256 _value) external
returns (bool success);

//transferFrom - once approved, it is used to transfer all or partial
allowed/approved
//tokens to spender
    function transferFrom(address _from, address _to, uint256 _value)
external returns (bool success);

//allowance -  it is to know the number of remaining approved tokens
    function allowance(address _owner, address _spender) external view
returns (uint256 remaining);

//trnasfer event - it  is used to log the transfer function activity
like from account, to account
//and how much token was transfered
    event Transfer(address indexed _from, address indexed _to, uint256
_value);

//approval event  - it is used inside approved function to log the
activity of approved function
    event Approval(address indexed _owner, address indexed _spender,
uint256 _value);

}

contract MyERC20Token is IERC20{

    mapping (address => uint256) public _balances;

    //Approval
```

```
    mapping (address => mapping(address => uint256)) _allowed;
    //1111 => 2222 - 10
    //1111 => 3333 - 20

    //name , symbol, decimal

    string public name = "BlkTraining";
    string public symbol = "BLTK";
    uint public decimals = 0;

    //uint256 - intial supply
    uint256 private _totalSupply;

    //address - creator's address
    address public _creator;

    constructor() public{
        _creator = msg.sender;
        _totalSupply = 50000;
        _balances[_creator] = _totalSupply;
    }

    function totalSupply() external view returns (uint256){
        return _totalSupply;
    }

    function balanceOf(address _owner) external view returns (uint256
balance){
        return _balances[_owner];
    }

    function transfer(address _to, uint256 _value) external returns
(bool success){
        require(_value > 0 && _balances[msg.sender] >= _value);

        _balances[_to] += _value;
        _balances[msg.sender] -=_value;

        emit Transfer(msg.sender, _to, _value);

        return true;


    }

    function approve(address _spender, uint256 _value) public returns
(bool success){
        require(_value > 0 && _balances[msg.sender] >= _value);

        _allowed[msg.sender][_spender] = _value;
```

```
        emit Approval(msg.sender,  _spender, _value);

        return true;

    }

     function transferFrom(address _from, address _to, uint256 _value)
external returns (bool success){
        require(_value > 0 && _balances[_from] >= _value &&
_allowed[_from][_to] >= _value);

        _balances[_to] += _value;
        _balances[_from] -=_value;

        _allowed[_from][_to] -= _value;

        return true;

     }

     function allowance(address _owner, address _spender) external
view returns (uint256 remaining){
         return _allowed[_owner][_spender];
     }
}
```